View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Hosei University Repository

法政大学学術機関リポジトリ
HOSEI UNIVERSITY REPOSITORY

パッキングされたマルウェアのバイナリーコード解析

著者	光吉 寛生
出版者	法政大学大学院情報科学研究科
雑誌名	法政大学大学院紀要．情報科学研究科編
巻	11
ページ	1-6
発行年	2016-03-24
URL	http://hdl.handle.net/10114/12236

パッキングされたマルウェアのバイナリコード解析
Packed Malware Analysis using Binary Code Distribution

光吉　寛生
Hiroo Mitsuyoshi
法政大学大学院情報科学研究科情報科学専攻
E-mail: hiroo.mitsuyoshi.8p@stu.hosei.ac.jp

Abstract*—Increase in Malware is a serious problem. Malware whose code is obfuscated by the software called packer may have a different binary pattern from normal malware. It requires unpacker, which is corresponding to the packer, to decode and classify it. It is important to detect the existence of packed malware without unpacking it to scan the mail on the server or files on the on-line storage. It is also important to detect the fact that the software is packed with some packer to deeply analyze the suspicious data. In this paper, two techniques for inferring malware and packer are proposed. First approach compares the rate of number of the incidence of the same binary code in difference malwares. Second approach treats a binary code as a vector and calculate the similarity between the pieces of it. Twenty-three specimens, which are generated from eight types of malware with three packers, are analyzed with these techniques. The statistics of analysis of packed malware is compared with the one of non-packed original malware. The result shows that the malware is identifiable under the packed status.

***Keywords*—malware, packing, N-gram**

1. はじめに

マルウェアによる攻撃は爆発的に増加する傾向にある．それらのマルウェアによる攻撃から自組織のネットワークを守るためには，IDS 等によるマルウェアの検出と速やかな種別の特定が重要である．IDS 等はインターネットから流れ込むパケットを監視し，2 種類の方法を用いてマルウェアの検知を行う．1 つが悪意のある通信パターンのシグネチャと類似しているかの確認を行い検知するシグネチャ型，もう一方が統計的に異常なパケットであるか確認を行い検知するアノマリ型である．既知のマルウェアの亜種が送り込まれた場合，シグネチャ型では検知できない可能性があるが，アノマリ型では検知できる可能性がある．しかし，アノマリ型では悪意のないパケットも誤検知してしまうといった問題点が存在する．

近年のマルウェアは，パッカーと呼ばれる難読化ツールにより暗号化や圧縮化，バイナリコードレベルの配置換えなどの難読化が施されている場合が多く，既知のマルウェアのシグネチャと比較を行うだけでは検知を行う事が出来なくなっている．難読化されることによって中身のコードを解析することも難しくなっている．そのようなパッキングされたマルウェアを解析にするためには，パッカーに対応した復元ツールであるアンパッカーが必要となる．また，通常のバイナリファイルやデータはパッキング処理が施されていないことから，パッキングされているかどうかを判別可能とするだけでも危険性の高いファイルを隔離するためには役に立つ．この観点では，パッキングされているファイルの検知，使用されているパッカーの種類を特定する技術も重要となっている．

既存手法では，パッキングされていないマルウェアに対して圧縮技術を応用することで種類を分別していた．この手法ではプログラム中に頻出する特徴的なバイナリパターンに対して圧縮が効くことを活用していると考えられる．これに対して，パッキングにより難読化やバイナリコードレベルの配置換えが行われると，この特性が失われる．そこで本研究では，パッキングされたままの状態のマルウェアでコード断片の出現頻度やコード類似性の比較を行うことで，パッキングされたマルウェアの種類やパッカーの種類を判別することを目指す．

2. 関連研究

この章では 2 つ関連研究を取り上げる．1 つは，N-gram と呼ばれる文字列検索手法を利用して悪意のあるバイナリコード検出を行う研究，もう 1 つは圧縮手法を利用してマルウェア分類を行う研究である．

2.1. N-gram によるコード解析

標準的なシグネチャベースのウィルス検出手法であると，被害をもたらした後にのみ，ウィルスを検出することが可能である．Abou-Assaleh らは新しい悪意のあるコードを検出するために N-gram を用いる手法

Supervisor: Prof. Toshio Hirotsu

を提案している[1]．N-gram とは文字列を検索するために，文字列をサイズ N 個の文字に分解し，分解した文字列を検索する手法である．ある文内の“マルウェア”という文字を検索する際に，N=2 とした場合，検索文字列を“マル”，“ルウ”，“ウェ”，“ェア”の 4 つの部分文字列に分解する．そしてこの 4 つ部分文字列を検索する事によって，検索を行いたい文字列を発見する手法である．既存のマルウェアから N-gram を用いてコードを抜き出し，コードとコードの出現頻度をまとめたプロファイルを作成する．プロファイルを用いてマルウェア間の距離を計算し，計算した距離と K 近傍法を用いてクラス分類を行う事で悪意のあるコードの検出に成功している．

2.2. 圧縮手法を用いたマルウェア分別

Tao Gong らは，コードのパターンによって圧縮率が異なることに着目したマルウェアの分別手法を提案している[2]．オペコードには機能的に類似している命令が複数あるため，そのような類似した命令を一つのコードに置き換えた中間表現へ変換する．また，処理列に着目するため，オペランドは除外する．そのようにしてできた実行バイナリの中間表現から PPM 圧縮モデルで作成する．パッキングされていないマルウェアに対して複数の圧縮モデルを適用し，高い圧縮率の得られたモデルにマルウェアを分類するというものである．PPM 圧縮モデルによって出現頻度の多いバイトパターンに圧縮がかかるようにしている．この手法では，圧縮に計算の対象としてパッキングされていないマルウェアを用いるため，パッキングされているものについては事前にアンパックすることが必要となる．パッキングされたマルウェアではオペコードが圧縮化や暗号化されているため中間表現への変換が出来ない事や，パッキングされたデータは圧縮化がすでに行われていて比較的圧縮が効きにくいことが想定される．本研究では圧縮ではなく，直接バイナリコードの解析を行う事でパッキングされたマルウェアの特定やパッキングの検知，パッカーの特定を目指す．

3. 本研究の目的

マルウェアがパッカーによってパッキングが行われると，マルウェア内でのコード配置やマルウェアの挙動が元々のマルウェアと変わってしまうため，パッキングされたマルウェアの検出は難しい．N-gram によるコード解析では，パッキングされているマルウェアに対しては考慮されていない．圧縮手法を用いたマルウェア分別では圧縮手法を用いるため，パッカーによる圧縮や暗号化が行われたマルウェアに対して本来の圧縮ができず，分別できないと考えられる．現在パッキングされたマルウェアを分類やパッカーの特定を行うためには，実際にマルウェアの実行を行いマルウェアの挙動をトレースし，その挙動より判断を行う必要となる．しかし，マルウェアの実行による特定は，特定の環境下においてのみ実行されるマルウェアに対しては有効でない可能性やマルウェアを実行するといったリスクが発生する．

パッキングされたままでマルウェアの特定やパッキングの検知やパッカーの特定を行う事が可能となれば，メールサーバー等に送られた疑わしいファイルを取り除く事が可能になると考えられる． そのため本研究ではパッキングされたままの状態のマルウェアでコード断片の出現頻度やコード類似性の比較を行い，パッキングマルウェアの特定やパッカーの特定が可能か解析を行った．パッキング方法には，様々な手法が考えられるが，本研究ではプログラムのコードの配置換えや圧縮化が行われているものと想定する．

N-gram を用いる事で，パッカーによってマルウェアがバイナリコードレベルの配置換えや，圧縮化されていたとしても，元のマルウェアのバイトコードの特徴や頻度を抽出できると考えられる． N-gram を用いる事によってパッカー特有のバイナリコード断片を見つけることが可能と思われる．圧縮化や変換によってコード自体が書き換えられてしまっていた場合においても，パッキング前のバイナリコードと書き換え後のバイナリコードの出現頻度は変化しないと推定できバイナリコードの出現頻度を用いて推測可能である．パッカーごとの圧縮化や変換の特徴の類似性が存在すると考えられるため，短いバイナリ列を見ることによって圧縮・変換手法の種類毎の特性が発見できるのではないか推測できる．対象を x86 アーキテクチャの実行可能ファイルとする．x86 アーキテクチャでは，オペランドの数がオペコードによって異なるため，バイナリコードレベルの配置換えされている場合にはオペコードとオペランドのセットを取り出すのは困難になる．そのためバイトコードを取り出す際に，前回取り出したバイトコードを含むように取り出す．そうすることによって命令の区切り目が不明であった場合においても，オペコードとオペランドをセットで抜き出すことが可能である．本研究では N-gram を用

表 1　コードの出現頻度による手法の実験(上表：良好，下表：不良)

		addware3				backdoor2				worm2			
		original	telock	Upack	yoda's	original	telock	Upack	yoda's	original	telock	Upack	yoda's
addware3	original		99.4	99.5	98.7	79.5	67.3	84.4	73.1	46.9	40.1	26.5	39.4
	telock	99.4		99.4	98.4	60.8	79.2	77.5	16.9	37.1	68.5	27.5	37.8
	Upack	99.4	99.4		98.9	80.4	76.8	90.0	76.9	14.6	19.8	27.7	18.4
	yoda's	98.8	98.6	99.1		74.8	26.2	81.0	76.4	38.4	43.1	29.0	67.2
backdoor2	original	69.9	39.2	68.8	63.4		97.8	96.8	96.2	28.8	22.6	21.4	20.1
	telock	43.6	58.3	44.0	11.7	97.6		98.5	94.4	19.6	62.6	15.8	18.9
	Upack	73.3	46.4	83.0	70.2	98.2	98.3		96.4	14.6	16.5	27.1	15.5
	yoda's	61.5	9.6	63.6	69.3	97.5	97.1	98.2		15.2	22.8	16.4	56.8
worm2	original	48.8	38.0	16.1	40.3	36.0	28.8	33.3	18.9		97.5	98.0	96.8
	telock	41.5	70.2	20.3	42.2	32.6	76.5	50.0	25.2	97.6		99.6	97.6
	Upack	28.2	29.9	30.0	32.6	44.9	42.0	72.2	29.1	97.8	99.7		98.6
	yoda's	41.1	40.8	18.4	68.1	33.3	31.2	58.1	59.4	97.2	98.3	98.9	

		addware1				backdoor1			worm1			
		original	telock	Upack	yoda's	original	upack	yoda's	original	telock	Upack	yoda's
addware1	original		96.2	99.6	99.2	70.3	42.9	34.7	87.6	69.1	84.6	46.9
	telock	41.9		76.2	33.0	50.2	42.9	58.7	68.1	90.1	100.0	85.7
	Upack	99.5	95.4		99.6	25.1	42.1	46.2	45.0	68.2	100.0	75.9
	yoda's	94.8	55.1	99.7		25.2	51.6	62.3	46.8	52.2	100.0	80.7
backdoo	original	43.3	49.6	26.9	35.7		99.4	98.1	85.6	71.9	75.0	50.4
	upack	14.0	31.3	28.5	64.9	98.7		99.3	75.8	88.9	100.0	92.7
	yoda's	6.3	69.4	65.0	59.9	95.6	99.6		65.0	63.5	100.0	89.4
worm1	original	49.9	73.0	49.1	27.1	75.6	74.6	60.8		73.9	93.3	80.9
	telock	33.7	91.2	39.5	29.6	63.0	80.0	59.8	75.6		100.0	96.9
	Upack	52.4	100.0	100.0	100.0	60.0	100.0	100.0	87.5	100.0		100.0
	yoda's	17.1	84.2	48.9	63.1	23.3	86.4	85.2	79.5	93.9	100.0	

いて，バイナリコードの出現頻度による手法とバイナリコードをベクトルとする手法の 2 つを提案する．

4. コードの出現頻度による手法

マルウェアがパッカーにパッキングされたとしてもマルウェア内のバイナリコードの出現頻度が変化しないバイナリコードもあるのではないかと考えた．そのため比較を行う 2 検体に含まれる同じバイナリコードの総和を母数として，同じバイナリコードで且つ同じ出現数の割合を計算する．元のマルウェアが同じでパッカーが異なった場合においては，マルウェアの内の特定のバイナリコードの出現数が同数になり割合が高くなると推測できる．元のマルウェアが異なり同じパッカーが用いられたマルウェア同士であれば，パッカーによって差し込まれたパッカー独自のバイナリコードの出現数が同数になると思われるため，他のパッカーでパッキングされた検体よりも高い割合になると考えられる．

4.1. 解析実験

本研究の実験に用いるマルウェアは 2 種類の方法を用いて，収集行った．1 つは，マルウェアと疑わしいファイルを配布している URL のリストを載せているウェブサイト[2]を利用し，載せられている URL からファイルをダウンロードする方法である．もう 1 つはマルウェアのサンプルを配布しているウェブサイト[4]からマルウェアをダウンロードする方法である．ウェブサイト[3]を利用した場合には，そのファイルがマルウェアであるか不明であるため，virus total [5]というウェブサービスを利用してマルウェアであるか判断を行う．virus total へファイルをアップロードしたファイルの各々スキャン結果を返す．そのスキャン結果からマルウェアであるか判断を行い，目的のマルウェアの探し出す．本研究では，addware 型，backdoor 型，worm 型の 3 種類を収集した．マルウェア 8 検体とパッカーを 3 種類用意し，各マルウェアを，各パッカーを用いてパッキングを行い，パッキングされたマルウェアの比較によって 2 つの手法を用いてマルウェアの推定が可能であるか確認を行う実験を行った．パッカーは，yoda’s Protecter 1.03.3，telock 0.98，Upack 0.399 を用いた．収集した検体 backdoor1 は，telock でパッキングを行う事が出来なかったため，元の検体と合わせて計 31 検体を用意した．

パッカーを用いてパッキングを行ったマルウェアに対して，読み込み位置を 2 バイトずつずらし，ファイルの最後まで 32 バイトずつ読み込む．読み込んだ値とマルウェア内での値の出現数を保存する．他のマルウェア内に，読み込んだ値と同じ値が含まれていた場合，総出現数と出現数が同じ値の総出現数を求める．その 2 つを用いて，出現数が同じ値の割合を計算した

表 2　双方のマルウェアに含まれていた同じバイナリコードが出現した数

		addware1				backdoor1			worm1			
		original	telock	Upack	yoda's	original	upack	yoda's	original	telock	Upack	yoda's
addware1	original		1247	33790	33720	11861	240	458	3318	149	13	245
	telock	2861		324	227	233	98	201	191	355	14	168
	Upack	33846	259		33566	283	228	145	60	22	635	29
	yoda's	35294	136	33512		325	190	509	111	92	17	305
backdoo	original	19262	236	264	230		7317	14287	2186	128	8	228
	upack	735	134	337	151	7369		7290	62	36	29	41
	yoda's	2507	170	103	529	14674	7272		177	159	9	425
worm1	original	5822	178	55	192	2474	63	189		226	30	235
	telock	306	351	38	162	146	40	169	221		19	256
	Upack	21	14	635	17	10	29	9	32	19		27
	yoda's	672	171	45	390	494	44	446	239	264	27	

表 3　提案手法を用いて計算した評価値の平均(上表)・分散(下表)

worm1_Upack	addware1	addware2	addware3	backdoor1	backdoor2	worm1	worm2	worm3
original	0.697	0.709	0.703	0.697	0.705	0.666	0.698	0.675
telock	0.698	0.709	0.672		0.671	0.670	0.671	0.676
Upack	0.704	0.715	0.677	0.696	0.676	1.000	0.669	0.676
yoda's pro	0.700	0.709	0.676	0.696	0.673	0.672	0.675	0.677
worm1_Upack	addware1	addware2	addware3	backdoor1	backdoor2	worm1	worm2	worm3
original	0.000864	0.000773	0.000725	0.000773	0.000732	0.001103	0.000700	0.000905
telock	0.000672	0.000628	0.000909		0.000898	0.000868	0.000887	0.000816
Upack	0.003233	0.002987	0.004024	0.000755	0.004085	0.000000	0.001102	0.001271
yoda's pro	0.000673	0.000630	0.000859	0.000684	0.000870	0.000851	0.000864	0.000802

のが表 1 となる．双方のマルウェアに出現したバイナリコードのそれぞれの出現数の総和を示したのが表 2 となる. 表 1, 表 2 共に縦軸は比較された検体であり，横軸の検体の値を元に数値を示している．

4.2. 解析結果

表 1 上表より，パッカーが違っていても元の検体が同じであれば，高い類似性が見られる．表 1 上表を見るとパッカーが同じであれば，他のパッカーでパッキングされた検体と比べて高い類似性が出ている事がわかる．表 1 下表を見ると同じマルウェアであっても必ずしも高い類似度となっている訳でない事がわかる．しかし，元々の検体と比較した検体との類似度が高くなっている. worm1 を Upack でパッキングされた検体は，元のマルウェアや使用されたパッカーに関係なく高い類似度が出ている事がわかる．表 2 から worm1 を Upack を用いてパッキングされた検体は，Upack を使用して addware1 から作成した検体を除き，出現数の総和が低い事が見られる．

4.3. 考察

ここまでで得られている実験の結果によると，パッカーが異なっても元々のマルウェアとの類似度は高くなっている．これは，パッカーを使用されても元々のマルウェア独自のバイナリコードが同頻度で残っていたためと考えられる．しかし，パッカーのアルゴリズムによっては残る独自のバイナリコードが異なるためにパッカーによっては高い類似性を示す訳ではない事がわかった. 表 1 下表では worm1 を Upack でパッキングされた検体が検体に関係なく高い類似度を示している原因は，表 2 を見ると双方に出現したバイナリコードが少ないことが起因しているではないかと考えれる．そのため検体に関係なく高い類似度を示す検体であるという事は表 2 から推測でき，実際に推定を行う際にはそのような検体を取り除くことによって推定が可能である．

5. コードベクトルによる手法

オペコードは似た機能を持つコマンドに対して，実際のバイナリのハミング距離が近くなる．バイナリコードの置き換えが発生し別のバイナリコードに置き換えられたとしても抜き出す事が可能とするために，本手法ではバイナリコードをベクトルと考え，類似度を計算することで似ているコードを探し出し，マルウェアの特定やパッカーの特定を行う手法である．

バイナリコードをベクトルとして見なして類似度の計算を行う事は，具体的には 16Byte のバイナリコードは，128bit であるため 128 次元のビットベクトル

表 4　各検体内でのバイナリコード位置

バイナリーコード/Upack	addware1	addware2	addware3	backdoor1	backdoor2	worm1	worm2	worm3
5a4d454b4e524c453233442e4c4c0000	0	0	0	0	0	0	0	0
4e524c453233442e4c4c00006f4c6461				4			4	4
4c4c00006f4c6461694c726272614179				12			12	12
694c726272614179000000065475074				20			20	20
726141790000000065475074f724163				24			24	24
8554ff0013163bc072c12bf4c3c108b1	520520	1813460	55124		29944	22408		
72c12bf4c3c108b116ff528db0047301	520528	1813468	55132		29952	22416		
c3c108b116ff528db0047301ff0bb016	520532	1813472	55136		29956	22420		51972
16ff528db0047301ff0bb0167309c105	520536	1813476	55140		29960	22424		51976
8d50822c56ff5d4cc50350c3038bc152	520556	1813496	55160		29980	22444		

バイナリーコード/telock	addware1	addware2	addware3	backdoor1	backdoor2	worm1	worm2	worm3
e800000000002b5e58c9027420cd51b9	623624	2019848	23048	–	29192	26632	33800	54792
00002b5e58c9027420cd51b900198b00	623628	2019852	23052	–	29196	26636	33804	54796
58c9027420cd51b900198b00f8c10273	623632	2019856	23056	–	29200	26640	33808	54800
00198b00f8c1027320cdc6838d338144	623640	2019864	23064	–	29208	26648	33816	54808
f8c1027320cdc6838d338144e8670002	623644	2019868	23068	–	29212	26652	33820	54812

バイナリーコード/yoda's protecter	addware1	addware2	addware3	backdoor1	backdoor2	worm1	worm2	worm3
00e800005d00ed81e2070040d58bc281	755532	2152268	79692	510796	38732	32588	55116	61772
5d00ed81e2070040d58bc281e2560040	755536	2152272	79696	510800	38736	32592	55120	61776
e2070040d58bc281e2560040e8520001	755540	2152276	79700	510804	38740	32596	55124	61780
d58bc281e2560040e85200010000c3c3	755544	2152280	79704	510808	38744	32600	55128	61784
03e80000eb00c2010ee80000e800ffd1	755560			510824			55144	

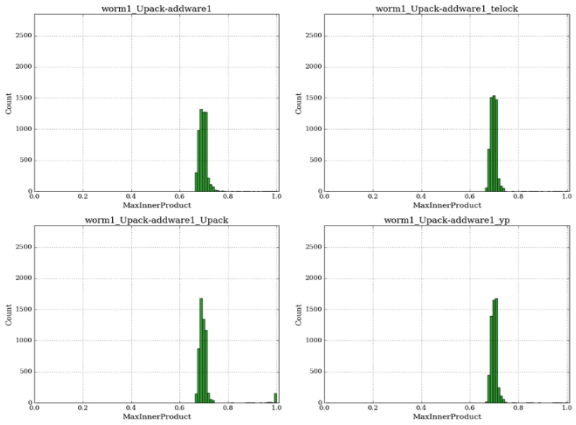

図 1　評価値のヒストグラム(左上:original,左下:Upack, 右上:telock,左下:yoda's protecter)

と考え，各ビットごとの排他的論理和の否定を求め，ビットの立っている数を数え，その数に抜き出したコードのビット数で割ったものを類似度の評価値とするものである．バイトコードの排他的論理和の否定を取る事で，バイトコードのビット表現が同じ部分を抜き出している．そのため似たバイナリコード同士であれば類似度の評価値は高くなる．

5.1. 解析実験

4 章 1 節と同じ検体を用いてコードベクトルによる手法を用いて解析実験を行った．検出したいマルウェア A と他のマルウェア B から N-gram を用いてバイナリコードを抜き出し，そのバイナリコードの評価値を計算した．その際にまったく関係性がないと考えられるバイナリコードとの評価値を取り除くため，マルウェア A 内の抜き出したバイナリコード一つに対してマルウェア B 内のバイナリコードのすべてとの評価値を計算した後に 1 に近い評価値を残す．マルウェア A 内のバイナリコードすべてに対して同様に計算し，評価値の平均と分散を計算する．抜き出したバイナリコードのサイズは 16Byte，抜き出す際の先頭位置の移動サイズを 4Byte とした．表 3 は Upack を用いて worm1 に対してパッキングを行った検体と他の検体との評価値の平均と分散を示した表である．表の横軸は元々の検体を表し，縦軸は用いたパッカーの種類である．図 1 は Upack を用いて worm1 から作成した検体と addware1 を用いて作成した検体との評価値のヒストグラムとなる．図 1 のヒストグラムのビンは 0.00 から 1.00 までの 0.01 刻みで設定し，横軸が評価値となり，縦軸がビンに属する評価値の個数となる．

5.2. 解析結果

表 3 上表より，予想と反して元が同一のマルウェアの平均が低くなった．平均はパッカーに依存せず，比較された検体の元のマルウェアによって平均値が変化しているように見られる．表 3 下表より比較した検体と同種のパッカーを用いられた検体は，他のパッカーでパッキングされた検体よりも標準偏差が高くなっている．図 1 左下のヒストグラムを見ると，0.99 から 1.00 の範囲のビンが高くなっている事がわかる．

5.3. 考察

表 3 上表の平均が他のマルウェアよりも小さくなったことに関しては, これは worm1 型の検体のサイズが他に比べて小さく，中に含まれているコードの種類が少ないために起こったのではないかと考えられる．そのためこの手法ではマルウェアの特定が難しいのではないかと考えられる．次に表 3 下表の標準偏差が同じパッカーでは高くなることに関しては，図 1 より 1.00 ビンが高くなっている事が関係しているのでないかと推測できる．これは評価値が 1 の組み合わせがあったことを示しており，同じパッカーであればまったく同じバイナリコードが含まれていた事を表している．そこでそのバイナリコードからパッカーの推定が可能ではないかと考え，同じバイナリコードを実際に抜き出してみることにした．表 4 は各パッカー内で出現したバイナリコードに対して先頭から何バイト目に出現しているかを示している．実験同様，16Byte ずつバイナリコードを抜き出し，4byte ずつ抜き出し位置ずらしている．抜き出したバイナリコードは，表 4 に記載していない他のパッカーでパッキングされた検体および元々の検体には一切含まれていない．表 4 を見るとパッカーによってパッキングされた検体において，パッカー独自のコードを持つ事が確認する事ができる．Upack でパッキングされた検体は，検体によって 2 種類のバイナリコードパターンが含まれている事が判明した．これらのバイナリコードは既存手法であるシグネチャ方式の検出行う事が可能ではないかと思われる．しかし，シグネチャ方式を回避するためパッカー独自のコードがマルウェア内に分散するようなパッカーが存在するもしくは今後作られると予想される．本手法では，パッカー独自のコードが分散していたとしても N のサイズを小さくすることによってパッカー独自のコードを見つけることが出来ると考えられる．以上より，N-gram を用いてバイナリコードを抜き出し，バイナリコードの評価値の分散を確認する事でどのパッカーが使用されたか，判別することが可能であることが判明した．

6. まとめ

本研究ではパッキングされたマルウェアの特定やパッキングの検知，パッカーの特定を目指して実際のマルウェアとパッカーを用いて N-gram を用いた手法を 2 つ提案し，実験を行った．

本論文ではコードの出現頻度による手法とコードベクトルによる手法の 2 つの手法を提案した．この 2 つの手法では N-gram を用いてバイナリコード抽出を行っており，パッカーによって難読化されたとしても，元のマルウェアのバイトコードの特徴や頻度を抽出できると考えられる． N-gram を用いる事によってパッカー特有のバイナリコード断片を見つけることが可能であると推測した．コードの出現頻度による手法では，マルウェアがパッカーによって難読化が行われていたしても，マルウェア内のバイナリコードの出現頻度が変化しないバイナリコードも存在するではないかと考えた．コードベクトルによる手法は，似た機能を持つオペコードはオペコード同士のハミング距離が近いといった特徴からバイナリコードをベクトルとして考え，バイトコードの排他的論理和の否定を取る事で似ているバイナリコードを抜き出し，その結果からパッキングされたマルウェアの分類やパッカーの特定が可能ではないかと考えた．

実際のマルウェア 8 種とパッカー3 種を収集し，元のマルウェアと合わせて計 31 種類の検体を作成した．作成した検体に対して 2 つの提案手法を用いてマルウェアの特定やパッカーの特定が可能であるかの実験を行った．結果 2 つの手法は，使用されたパッカーを推定することが可能であることが判明した．提案手法を用いる事によってパッキングされたマルウェアの解析を行う際に，重要な情報となるパッカーの情報を得られることができる．コードの出現頻度による手法では，パッキングされたマルウェアの特定が可能である事が判明した．

参考文献

[1] Tony Abou-Assaleh, Nick Cercone, Vlado Keˇselj, Ray Sweidan “N-gram-based Detection of New Malicious Code” Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International

[2] Tao Gong, Xiaobin Tan, Ming Zhu,”Malware Detection via Classifying With Compression” Information Science and Engineering (ICISE), 2009 1st International Conference on

[3] 「The VirusWatch Archives」http://lists.clean-mx.com/pipermail/viruswatch/ (参照 2016-2-1)

[4]「Open Malware」 http://www.offensivecomputing.net/ (参照 2016-2-1)

[5]「virustotal」https://www.virustotal.com/ja/(参照 2016-2-1)

[6]「X86 Opcode and Instruction Reference」 http://ref.x86asm.net/ (参照 2016-2-1)